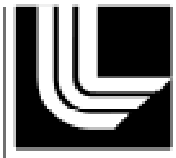

LAWRENCE LIVERMORE NATIONAL LABORATORY

LLNL-TR-650617

HPC4Energy Final Report GE Energy

Steven G. Smith, LLNL
Devin T. Van Zandt, GE Energy Consulting
Brian Thomas, GE Energy Consulting
Dr. Sajjad Mahmood, GE Energy Consulting
Dr. Carol S. Woodward, LLNL

May 2013

Contact:
Steven Smith
Lawrence Livermore National Laboratory
7000 East Avenue, L-561
(925) 423-8958
smith84@llnl.gov

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Executive Summary

Power System planning tools are being used today to simulate systems that are far larger and more complex than just a few years ago. Advances in renewable technologies and more pervasive control technology are driving planning engineers to analyze an increasing number of scenarios and system models with much more detailed network representations. Although the speed of individual CPU's has increased roughly according to Moore's Law, the requirements for advanced models, increased system sizes, and larger sensitivities have outstripped CPU performance. This computational dilemma has reached a critical point and the industry needs to develop the technology to accurately model the power system of the future. The hpc4energy incubator program provided a unique opportunity to leverage the HPC resources available to LLNL and the power systems domain expertise of GE Energy to enhance the GE Concorda PSLF software. Well over 500 users worldwide, including all of the major California electric utilities, rely on Concorda PSLF software for their power flow and dynamics. This pilot project demonstrated that the GE Concorda PSLF software can perform contingency analysis in a massively parallel environment to significantly reduce the time to results. An analysis with 4,127 contingencies that would take 24 days on a single core was reduced to 24 minutes when run on 4,217 cores. A secondary goal of this project was to develop and test modeling techniques that will expand the computational capability of PSLF to efficiently deal with systems sizes greater than 150,000 buses. Toward this goal the matrix reordering implementation time was sped up 9.5 times by optimizing the code and introducing threading.

Contents

Abstract

Power System planning tools are being used today to simulate systems that are far larger and more complex than just a few years ago. Advances in renewable technologies and more pervasive control technology are driving planning engineers to analyze an increasing number of scenarios and system models with much more detailed network representations. Although the speed of individual CPU's has increased roughly according to Moore's Law, the requirements for advanced models, increased system sizes, and larger sensitivities have outstripped CPU performance. This computational dilemma has reached a critical point and the industry needs to develop the technology to accurately model the power system of the future. The HPC4Energy Innovation provided a unique opportunity to leverage the HPC resources available to LLNL and the power systems domain expertise of GE Energy. This project demonstrated that the GE Concorda PSLF software can perform contingency analysis in a massively parallel environment to significantly reduce the time to results. We show that an analysis with 4,127 contingencies that would take 24 days on a single core was successfully scaled to 4,127 cores and the solution time was reduced to 24 minutes. We also analyzed and improved the capability of PSLF to efficiently deal with systems sizes greater than 150,000 buses by speeding up the matrix reordering implementation by 9.5 times using a combination of code optimization and introducing threading.

Introduction

For nearly a century, a core group of GE technical experts has focused its energies on solving the electric power industry's most pressing challenges-driving the evolution of electric power systems with greater affordability, reliability, and efficiency. Today, GE Energy Consulting provides innovative solutions across the entire spectrum of power generation, delivery, and utilization. By developing and customizing the GE Concorda PSLF software (PSLF), GE software experts provide clients a comprehensive set of state-of-the-art tools to assess the economic and technical performance of interconnected power systems. Well over 500 users worldwide, including all of the major California electric utilities, rely on Concorda PSLF software for their everyday power flow and dynamics needs (www.ge-concorda.com).

At a very high level, PSLF is a software tool that enables power system planners to perform (AC and DC) steady state power flow and dynamics analysis. PSLF is the standard tool used in all of WECC (www.wecc.biz). Planners using PSLF develop a detailed bus-branch model of the power system and PSLF calculates the voltages and angles at every bus (up to 80,000) in the network model using a Newton-Raphson solution methodology. In addition to the steady state calculation of voltage and angle, PSLF performs dynamics analysis that represents the dynamic performance under system disturbances. It adds comprehensive dynamic modeling of generators, excitation controls, turbines, governors, driven loads, relays, and other system components whose dynamics are significant in the 0-30 radian/second bandwidth. Both steady state and dynamic simulations are performed on a regular basis by PSLF users.

Power System planners across the globe assess system reliability by performing Contingency Analysis. In North America, NERC has standards that define and classify types of “contingencies” that planners are required to consider and analyze on an annual basis. A contingency is essentially an outage of a component or components in the power system. As expected, outage of one element can have an impact on the flow of power in a large interconnected power system. A complete contingency analysis involves the outage of combinations of equipment and re-evaluating the power flow solution results for each set of outage. If there is a line that exceeds its rating due to a particular outage, then this outage may be of concern to the utility owning the line. Typically, the industry uses the “N-x” nomenclature to define a contingency where x is the number of elements to be outaged for each event. For example, if x=1, then the contingency analysis will outage every element in a system one at a time and re-solve the power flow. With x=2, two elements would be outage each event. As x increases, the severity of the contingency event increases.

In practical terms, the number of contingencies performed by PSLF users has increased significantly in the last couple of years. We have seen customers running greater than 50,000 contingencies on a single CPU and taking days to complete. Since each contingency is independent of the other contingencies, this type of analysis is easily deployed on an HPC architecture. The first major task of the project was enabling a contingency analysis to run in parallel.

For today’s system sizes, PSLF performs most of its steady state and dynamics simulations very quickly (typically less than 1 minute). Therefore, the time to complete a contingency analysis is largely dictated by the total number of contingencies. As system complexity increases to include new renewable generation sources and more buses, the simulation runtimes will become more of a factor in the overall contingency runtimes. LLNL and GE staff worked together to profile PSLF and examined key functions for further optimization. Two major algorithms were examined, the linear solver and a matrix reordering step. Due to project time constraints only the matrix reordering function was optimized.

Porting PSLF to Linux

The first step towards running PSLF in an HPC environment was porting PSLF to the Linux operating system from the current shipping target of MS Windows. The target machine was the LLNL Sierra system which is a Linux cluster configured for large parallel jobs. It has 1,856 batch nodes with 2 Intel Xeon EP X5660 CPU’s with 6 cores each. Each node has 24 GB of memory. Nodes are connected with an InfiniBand QDR high-speed interconnect. Sierra runs the Clustered High Availability Operating System (CHAOS) Linux variant of RedHat 6.2 used on the LLNL HPC machines [CHAOS(2010)]. However, nothing in the port was specific to this Linux distribution. In previous releases PSLF had been run on Unix variants so the porting effort for system calls was not difficult and simply involved re-enabling and updating OS specific calls. A Dell Precision M6700 mobile workstation with a Intel Core i7 Q 820 running Windows 7 was also utilized in this work for verification and performance studies.

The more difficult problem was transitioning the code from a 32 bit architecture to a 64 bit architecture. Size assumptions were embedded in data structures, in particular several data structures being written as binary objects to files. A thin abstraction layer was introduced for data types to hide 32/64 architecture dependent lengths from the application. PSLF uses a Java based GUI front-end and Java is used as the main program driver for the command line version. The Java code and JNI wrapper code was ported without difficulties from the Windows version. As is expected with Java we did not need to recompile the Java portion of the application and we were able to use the byte code generated for the Windows version without modification.

Parallel contingency analysis

As previously discussed a common use of PSLF is to run hundreds of contingencies and a future goal is to run thousands. This is an “embarrassingly parallel” problem as each contingency does not contain any dependencies on the others. In order to avoid code divergence from the main PSLF codebase the contingency parallelization was implemented as a set of scripts to run the existing base PSLF code in parallel. This approach was taken over a more intrusive modification of inserting the parallelism inside the main PSLF code. A set of Python and C codes was developed to split apart existing PSLF input files into individual files for each contingency, execute each contingency on the nodes of the parallel machine, and finally merge the results back into a single output file. Some variance was introduced in the console output the user sees but the solution results are identical. The ordering of results was preserved, if this ordering restriction was removed additional parallelization could be done to the post processing step. The execution of the parallel contingencies is done using a single MPI job with contingencies being executed on each MPI Task within that job. This methodology was chosen since the schedulers on LLNL machines are targeted at large jobs and individually scheduling thousands of single node tasks is not permitted by local policies.

An initial static load balancer in which contingencies are statically allocated to MPI tasks at program launch was implemented first. Studies were conducted using a 63K bus problem and a larger 73K bus problem. As seen in Figure 1 the static load balancer suffered from large load imbalances which caused large deviations in the scaling results if several outliers were scheduled on the same core. Figure 2 shows a histogram of the runtimes for the 62K bus contingency study. While the vast majority of contingency runtimes was 8 to 9 minutes, the range was from 2 to 20 minutes. Note that the total runtime will be limited by the longest running contingency. To address the load imbalance the static load balancer was replaced with master-slave scheme which removed the large imbalances. Future work would look at employing scalable load balancers, such as a parallel work-stealing approach. Even with these simple strategies the speedup obtained was reasonable and the master process was not observed to be a severe bottleneck during this study.

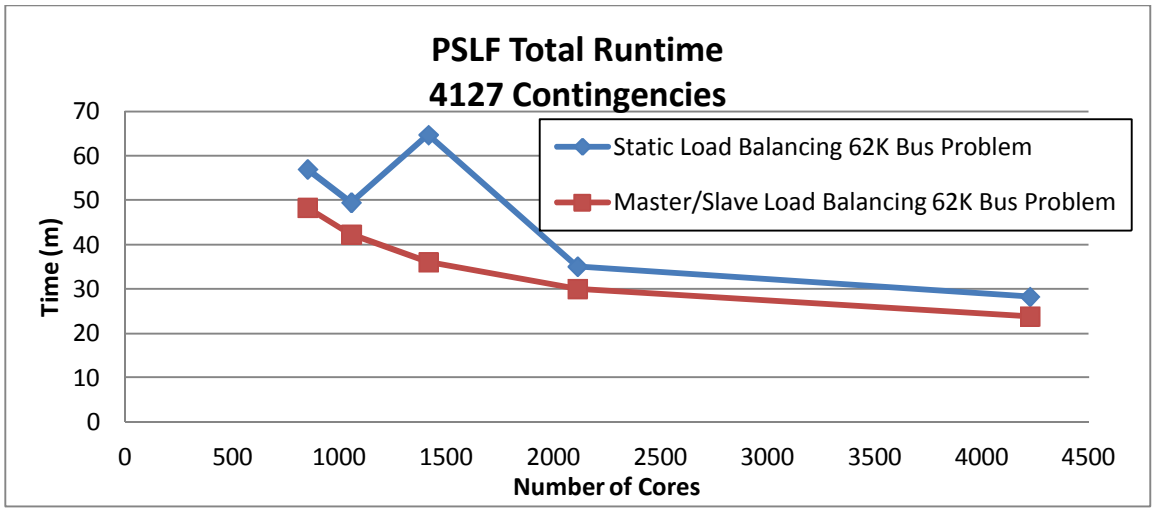

Figure 1
Scaling results of PSLF for 4127 contingencies on 62K bus problem

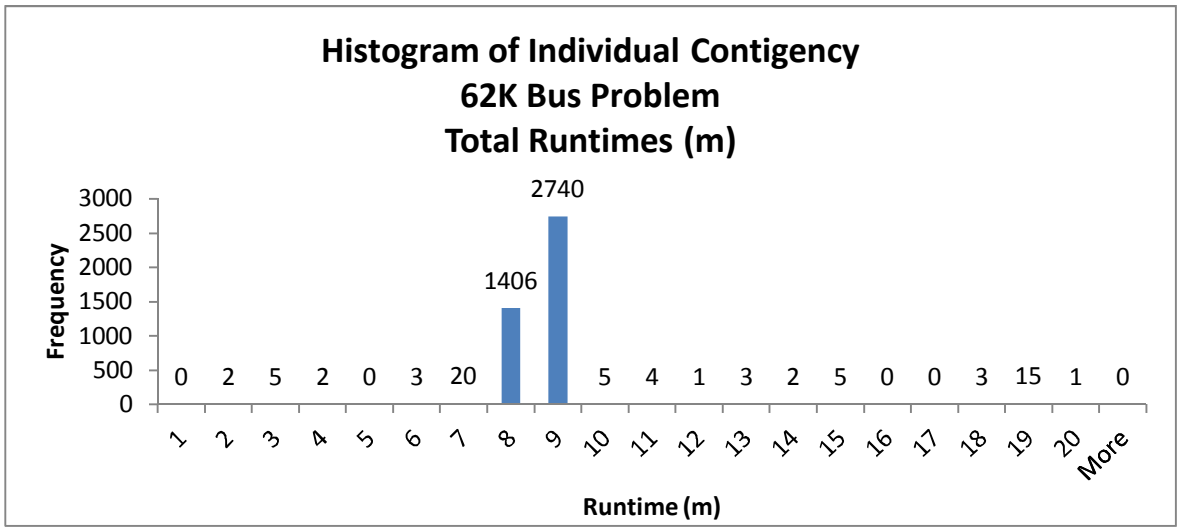

Figure 2
Histogram of individual contingency runtimes, frequency numbers indicate number of contingencies that took the indicated number of minutes to complete.

Profiling and optimization

After the initial effort of parallelizing the contingency analysis the focused turned to analyzing the performance of the execution of a single contingency. The goal was to speedup the processing of each contingency and make an initial step towards enabling larger problems. First the code was profiled to determine which methods should be considered targets for further optimization. A matrix reordering step was determined to be a key performance bottleneck and was chosen as the focus for the optimization work.

As a preliminary optimization step we compared C compilers using Intel version 13.0.1 and GNU version 4.4.6 compilers. The Intel compiler does a significantly better job at optimizing the matrix reordering over the GNU GCC compiler. These are results with optimization with the –O3 flag on both compilers. Additional more specific optimization flags, such as –fast or processor specific optimization flags, did not significantly affect performance. This resulted in a speedup of 2.3 for the method under Linux. After examining the reordering code it was noticed that several data structures where using data types larger than necessary for the values being stored, for example an integer for a Boolean value. Modifying the data structure and several other optimizations yielded a speedup of 7 under Windows and 1.7 under Linux. The final optimization step was to parallelize the looping constructs using OpenMP. Each loop was examined and appropriate OpenMP pragmas were added. This resulted in a speedup of 1.3 on Windows and 1.8 under Linux. As can be seen in Figure 5 for this size of problem parallelizing beyond 4-6 cores did not significantly improve performance. The parallelism changes were limited to modifications that would recreate the exact same behavior as the original sequential version with the exception that some error messages may appear in a different order. This enabled error checking to be done in parallel at a loss of deterministic ordering when multiple failure conditions are found. This allowed non-determinism does not impact the reordering results. With all optimizations applied we were able to gain a speedup of 9.6 times over the original code under Windows and a speedup of 7.2 times under Linux.

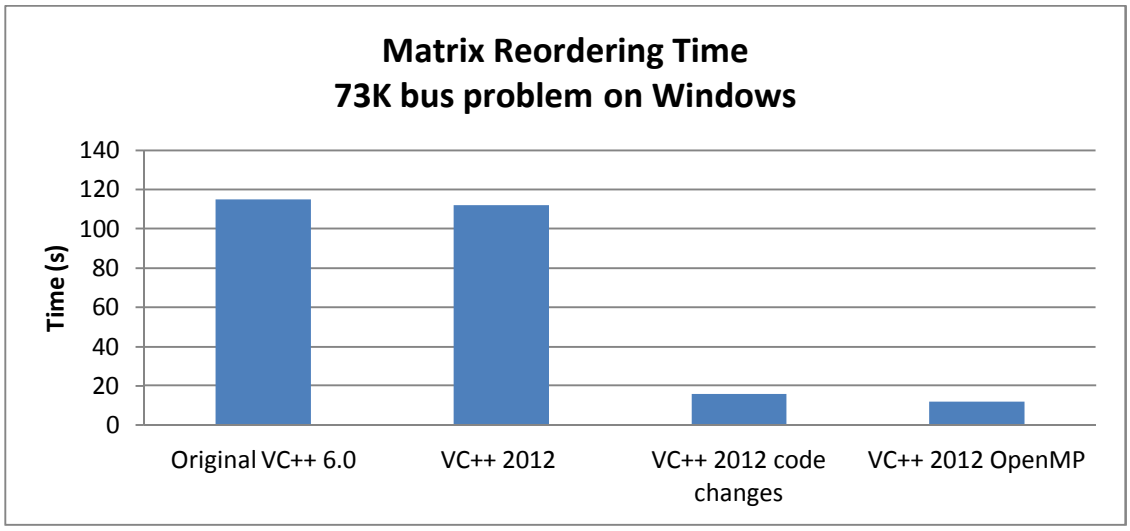

Figure 3
Matrix reordering time on 73K bus problem running under Windows

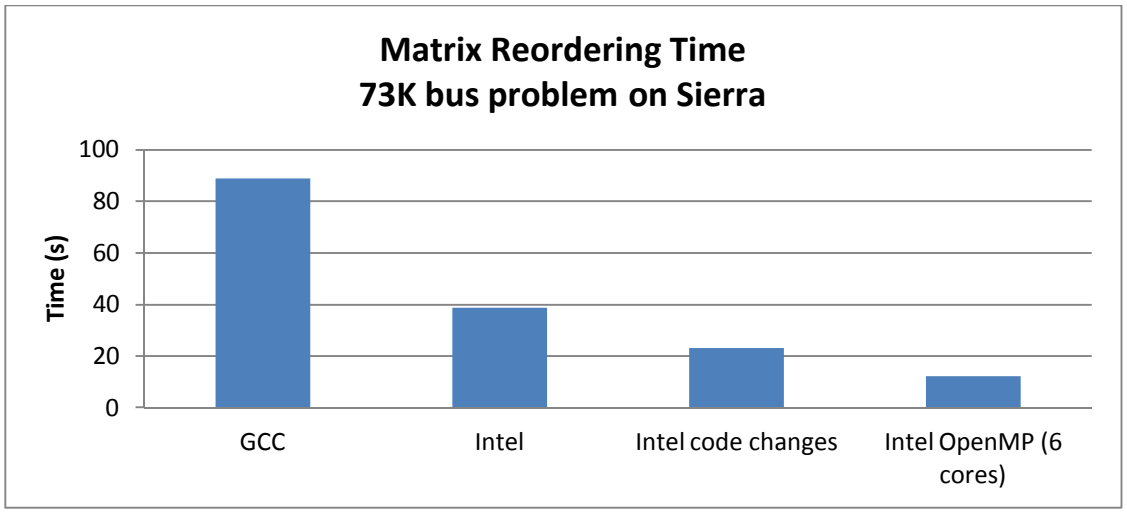

Figure 4
Matrix reordering time on 73K bus problem running on Sierra

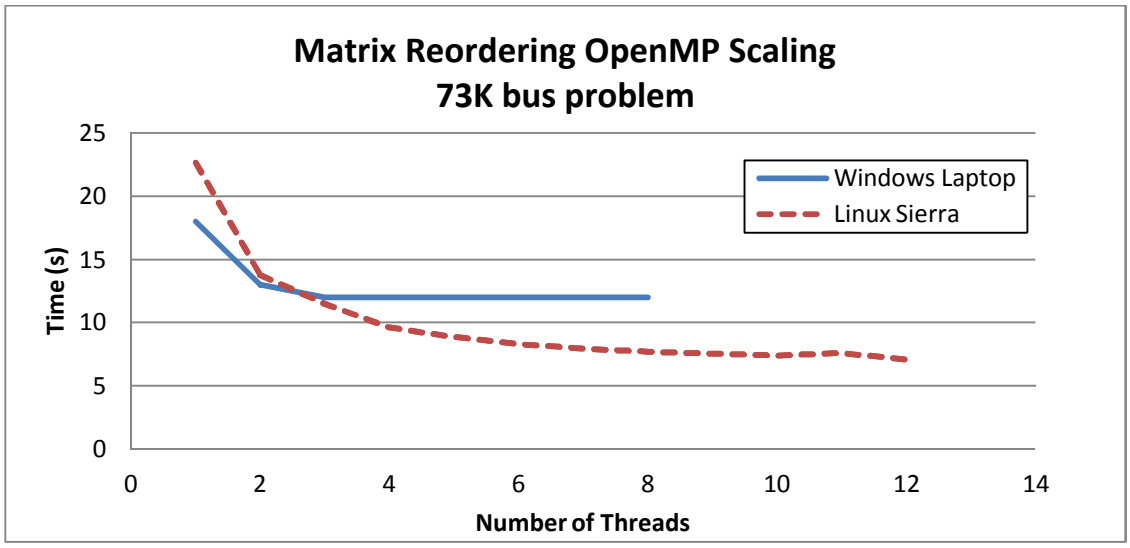

Figure 5
Matrix reordering scaling results for OpenMP

Figure 6 is a final scaling study conducted on the 73K bus problem which includes all optimizations described in the preceding section. A single OpenMP threads was used as running a single contingency per core results in better throughput than can currently be achieved by the limited threading that was introduced into PSLF. Figure 7 is a plot of the parallel efficiency. Parallel efficiency is defined as [Fox:1998],

$$E_p = \frac{T_1}{pT_p},$$

Note that the parallel efficiency is limited by the longest single contingency so efficiency at very large processor counts will be constrained. Most processors will be idle waiting for the longest running contingency to finish. To scale to larger numbers of cores, a single contingency solve would have to parallelized. A single core would have required an estimated 24.1 days to run all of the contingencies. These results demonstrate that a modestly sized cluster of 32 nodes reduces this time to far more manageable 95 minutes.

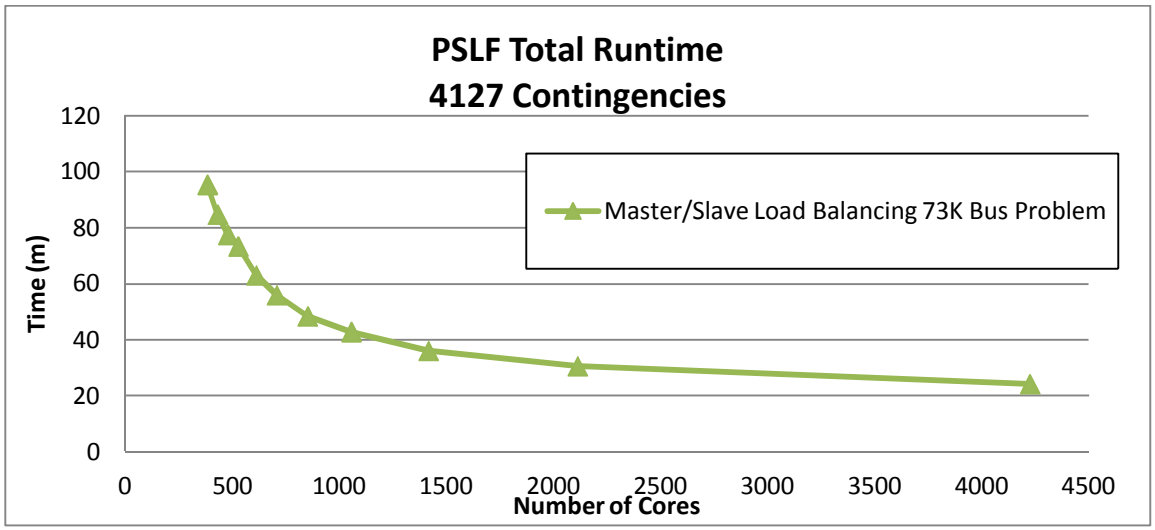

Figure 6
Scaling results of PSLF for 4127 contingencies on 72K bus problem

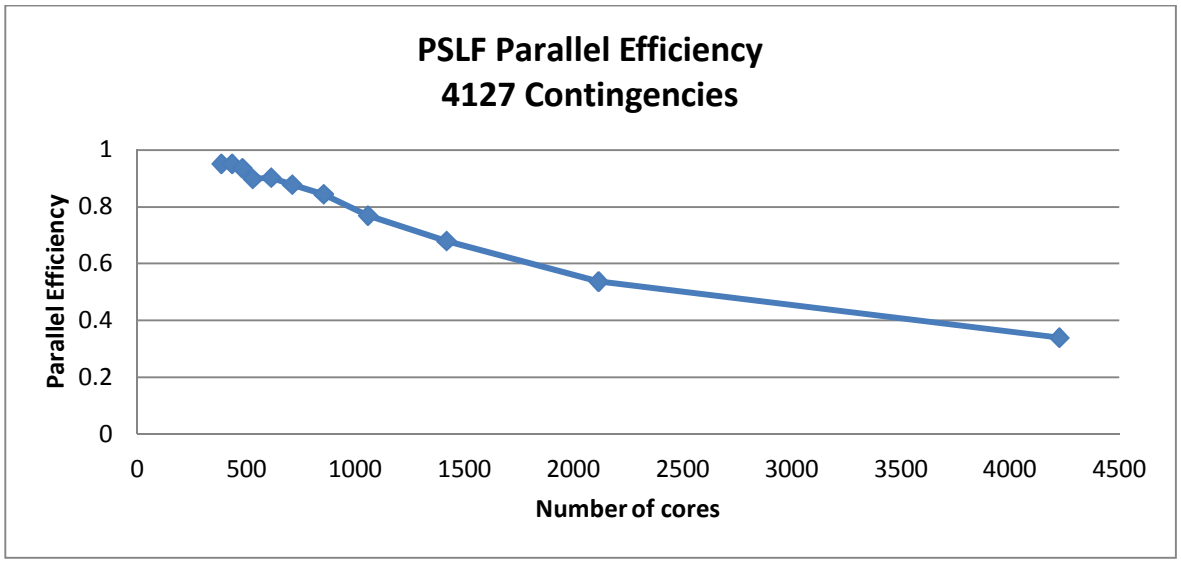

Figure 7
Parallel efficiency of PSLF for 4127 contingencies on 72K bus problem

Testing and Verification

The sequential version of PSLF was taken as the target design requirement so the sequential version results were used as the major point of comparison for verification of the HPC results. Test cases were run the existing Windows version and the new Linux port to verify that that the solutions were consistent with prior solutions. Outputs were compared to ensure that values such as the voltage magnitude and angle were within solver tolerances.

The matrix reordering changes were also verified using the existing code base as the target behavior. The existing algorithm and new algorithm were instrumented to record each reordering that was done to the matrix and we verified that the exact same sequence of reordering steps was done by both algorithms for several sample problems. The OpenMP parallelism was tested by running on 1 to 12 threads and comparing results to the output precision. We also used Inspector XE 2013 utility from Intel to analyze the added code for potential race conditions [Intel-2013].

Benefits to Company and LLNL

Successful deployment of PSLF in a massively parallel HPC environment and improvements to the computational performance of PSLF will benefit the GE Energy in a number of ways:

- Significant improvements in PSLF time to results enables system planners to focus more time on results analysis and less time on performing analysis. This should lead to an improved ability to identify potential major system reliability concerns.
- Runtime reductions should improve overall system planner efficiency. This has a direct financial benefit for utilities and consultants that use PSLF for contingency analysis.
- Expanding PSLF computational capabilities to more easily evaluate new emerging clean energy technologies and promote the technical benefits of these technologies.
- Enhancing PSLF scalability to model more detailed network models greater than 150,000 buses will enable system planners to more accurately represent real world conditions and perform better system event diagnosis.
- Improvements to PSLF computational engine will help to improve our competitiveness against other similar software packages, such as DigSilent (www.digsilent.de).

LLNL benefited in a number of ways.

- Demonstrated how LLNL HPC expertise can be applied to an industry application to significantly improve performance.
- Enhanced knowledge of power system modeling through working with GE's domain experts.

- Enhanced understanding of level of effort required in porting Windows based workstation applications to an HPC environment.
- Experience running thousands of instances of an application, traditional LLNL applications run 100's of large instances. Experience is being fed into future job schedulers and scheduling policies.

Conclusions

The project successfully demonstrated the feasibility of running PSLF in an HPC environment to decrease time to solution for a large contingency study from 24 days to 24 minutes. In additional to running parallel contingencies the project also successfully applied code optimization techniques to improve a key algorithm in PSLF. A speedup of 9.6 times was achieved for Windows and 7.2 times for Linux. These enhancements will enable system planners to focus more time on results analysis and less time on performing analysis. This should lead to an improved ability to identify potential major system reliability concerns.

References

[CHAOS:2010], CHAOS – Clustered High Availability Operating System. https://computing.llnl.gov/linux/projects.html

[Fox:1998] G. Fox, M. Johnson, G. Lyzenga, S. Otto, J. Salmon, AND D. Walker, Solving Problems on Concurrent Processors, Volume 1, Prentice-Hall, Englewood Cliffs, NJ.

[Intel:2013], Intel® Inspector XE 2013. http://software.intel.com/en-us/intel-inspector-xe